INTERNATIONAL STANDARD

ISO/IEC 14496-2

Second edition
2001-12-01

AMENDMENT 2
2002-02-01

Information technology — Coding of audio-visual objects —

Part 2:
Visual

AMENDMENT 2: Streaming video profile

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

AMENDEMENT 2: Cours du profil vidéo

Reference number
ISO/IEC 14496-2:2001/Amd.2:2002(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this Amendment may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to International Standard ISO/IEC 14496-2:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Coding of audio-visual objects — Part 2: Visual

AMENDMENT 2: Streaming video profile

1) Add the following text to the end of ‘Purpose’ of ‘Introduction’:

“

Two profiles are developed in response to the growing need for a video coding method for Streaming Video on Internet applications. It provides the definition and description of Advanced Simple (AS) Profile and Fine Granularity Scalable (FGS) Profile. AS Profile provides the capability to distribute single-layer frame based video at a wide range of bit rates available for the distribution of video on Internet. FGS Profile uses AS Video Object in the base layer and provides the description of two enhancement layer types - Fine Granularity Scalability (FGS) and FGS Temporal Scalability (FGST). FGS Profile allows the coverage of a wide range of bit rates for the distribution of video on Internet with the flexibility of using multiple layers, where there is a wide range of bandwidth variation.

“

2) Add the following text into ‘Introduction’ following ‘Error Resilience’:

“

Fine Granularity Scalability

Fine Granularity Scalability (FGS) provides quality scalability for each VOP. Figure AMD2-1 shows a basic FGS decoder structure.

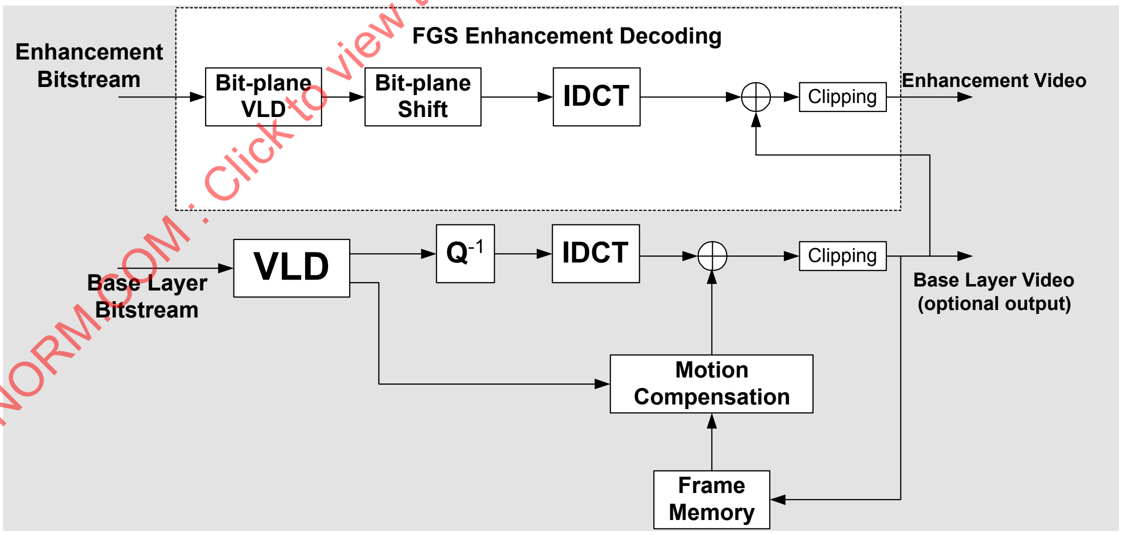

Figure AMD2-1 — A Basic FGS Decoder Structure

To reconstruct the enhanced VOP, the enhancement bitstream is first decoded using bit-plane VLD. The decoded block-bps are used to reconstruct the DCT coefficients in the DCT domain which are then right-shifted based on the frequency weighting and selective enhancement shifting factors. The output of bit-plane shift is the DCT coefficients of the image domain residues. After the IDCT, the image domain residues are reconstructed. They are added to the reconstructed clipped base-layer pixels to reconstruct the enhanced VOP. The reconstructed enhanced VOP pixels are limited into the value range between 0 and 255 by the clipping unit in the enhancement layer to generate the final enhanced video. The reconstructed base layer video is available as an optional output since each base layer reconstructed VOP needs to be stored in the frame buffer for motion compensation.

The basic FGS enhancement layer consists of FGS VOPs that enhance the quality of the base-layer VOPs as shown in Figure AMD2-2.

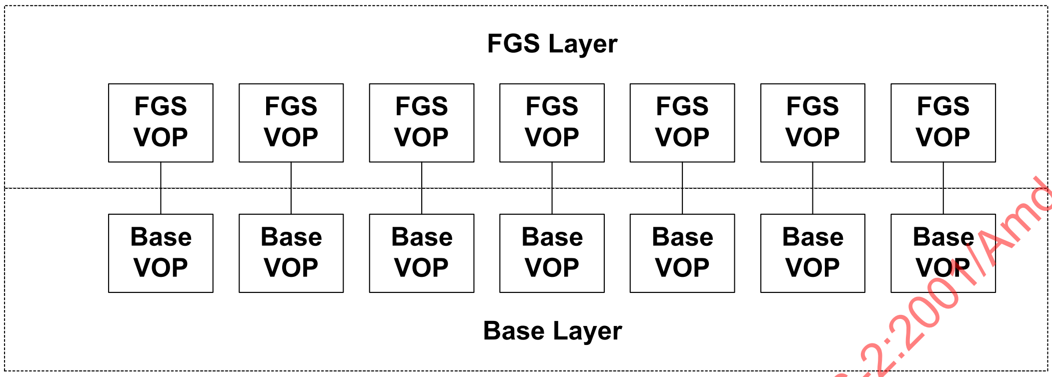

Figure AMD2-2 — Basic FGS Enhancement Structure

When FGS temporal scalability (FGST) is used, there are two possible enhancement structures. One structure is to have two separate enhancement layers for FGS and FGST as shown in Figure AMD2-3 and the other structure is to have one combined enhancement layer for FGS and FGST as shown in Figure AMD2-4.

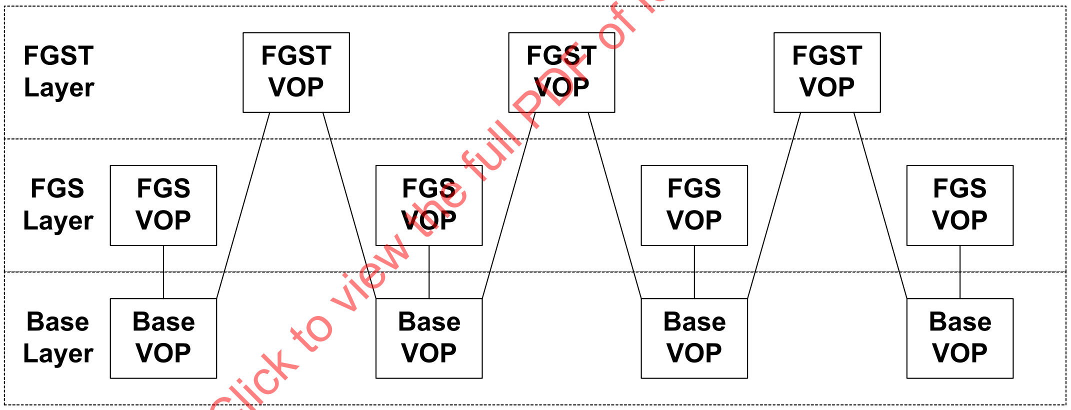

Figure AMD2-3 — Two Separate Enhancement Layers for FGS and FGST

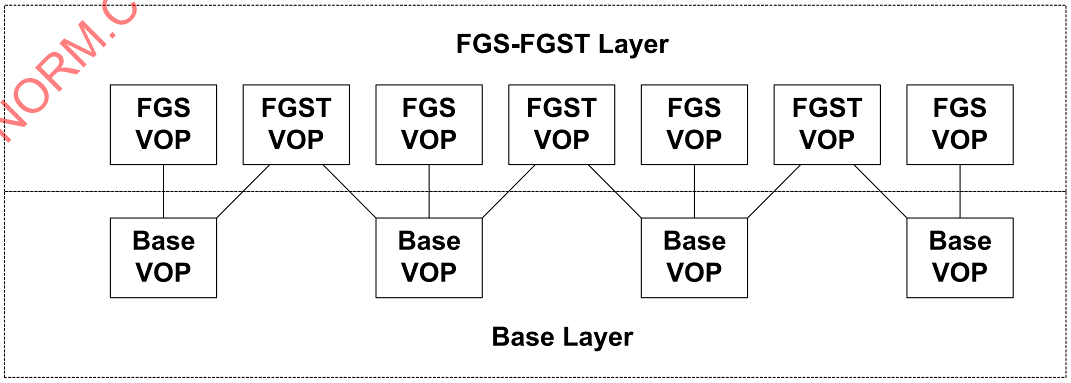

Figure AMD2-4 — One Combined Enhancement Layer for FGS and FGST

In either one of these two structures that include FGS temporal scalability, the prediction for the FGS temporal scalable VOPs can only be from the base layer. Each FGS temporal scalable VOP has two separate parts. The first part contains motion vector data and the second part contains the DCT texture data. The syntax for the first part is similar to that in the temporal scalability described in subclause 6.2. The DCT texture data in the second part are coded using bit-plane coding in the same way as that in FGS. To distinguish the temporal scalability in subclause 6.2 and FGS temporal scalability, the FGS temporal scalability layer in Figure AMD2-3 is called “FGST layer”. The combined FGS and FGST layer in Figure AMD2-4 is called “FGS-FGST layer”. The “FGS VOP” shown in Figure AMD2-3 and Figure AMD2-4 is an fgs vop with **fgs_vop_coding_type** being ‘I’. The “FGST VOP” shown in Figure AMD2-3 and Figure AMD2-4 is an fgs vop with **fgs_vop_coding_type** being ‘P’ or ‘B’.

The code value of **profile_and_level_indication** in VisualObjectSequence() has been extended to include the profile and level indications for AS Profile and FGS Profile. The identifier for an enhancement layer is the syntax **video_object_type_indication** in VideoObjectLayer(). A unique code is defined for FGS Object Type to indicate that this VOL contains fgs vops. Another unique code is defined for AS Object Type to indicate that this VOL is the base-layer. There is a syntax **fgs_layer_type** in VideoObjectLayer() to indicate whether this VOL is an FGS layer as shown in Figure AMD2-2 and Figure AMD2-3, or an FGST layer as shown in Figure AMD2-3, or an FGS-FGST layer as shown in Figure AMD2-4. Similar to the syntax structure in subclause 6.2, under each VOL for FGS, there is a hierarchy of fgs vop, fgs macroblock, and fgs block. An fgs vop starts with a unique **fgs_vop_start_code**. Within each fgs vop, there are multiple vop-bps. Each vop-bp in an fgs vop starts with an **fgs_bp_start_code** whose last 5 bits indicate the ID of the vop-bp. In each fgs macroblock, there are 4 block-bps for the luminance component (Y), 2 block-bps for the two chrominance components (U and V) for the 4:2:0 chrominance format. Each block-bp is coded by VLC.

“

3) Add the following subclauses in clause 3

"

3.AMD2.1 **block-bp**: An array of 64 bits, one from each DCT coefficient at the same significant position of accuracy in a zigzag scan order. When frequency weighting is used, block-bps are formed after the weighting is applied to the DCT coefficients in an 8x8 block.

3.AMD2.2 **end of plane; eop**: A symbol to indicate whether a ‘1’ bit is the last 1’ bit of a block-bp.

3.AMD2.3 **fgs block**: An 8-row by 8-column matrix of bits, each from one DCT coefficient at the same significant position of accuracy, or its coded representation. The usage is clear from the context.

3.AMD2.4 **fgs macroblock**: The four block-bps of luminance component (Y) and the two (for 4:2:0 chrominance format) corresponding block-bps of chrominance components (U and V) with the same accuracy significance coming from the DCT coefficients of a macroblock. It may also be used to refer to the coded representation of the six block-bps. The usage is clear from the context.

3.AMD2.5 **fgs macroblock number**: A number for an fgs macroblock within a vop-bp. The fgs macroblock number of the top-left fgs macroblock in each vop-bp shall be zero. The fgs macroblock number increments from left to right and from top to bottom.

3.AMD2.6 **fgs run**: The number of ‘0’ bits preceding a ‘1’ bit within a block-bp.

3.AMD2.7 **fgs temporal scalability; FGST**: A type of scalability where an enhancement layer uses predictions from sample data derived from the base layer using motion vectors. The VOP size in the enhancement layer is the same as that in the base layer. FGST is a specific type of temporal scalability where all DCT coefficients are coded using bit-plane coding as in FGS.

3.AMD2.8 **fgs vop**: The pixel differences between the original VOP and the reconstructed VOP in the base layer. It may be used to refer to the DCT coefficients of the pixel differences or the original VOP. It may also be used to refer to the coded representation of the DCT coefficients. In the context of FGST, fgs vop refers to the original temporal scalable VOP. The usage is clear from the context.

3.AMD2.9 **fine granularity scalability; FGS**: A type of scalability where an enhancement layer uses prediction from sample data of reconstructed VOP in the base layer. The encoded bitstream for each fgs vop can be truncated into any number of bits. The truncated bitstream for each fgs vop can be decoded to provide quality enhancement proportional to the amount of bits in the truncated bitstream of the fgs vop. The fgs vop has the same size and VOP rate as those of the base layer.

3.AMD2.10 **vop-bp**: An array of block-bps with the same accuracy significance in an fgs vop. There are three color components (Y, U, and V) in a vop-bp. Each color component in a vop-bp consists of all the block-bps of that color.

"

4) Add the following subclause to subclause 5.2:

"

Definition of start_of_bit_plane() function

The function start_of_bit_plane() returns 1 if the next bit in the bitstream is the first bit of the codes associated with a vop-bp. Otherwise it returns 0.

"

5) Add the following text to the end of subclause 6.1:

"

In a typical application of FGS, the bitstream at the input of an FGS decoder is a truncated version of the bitstream at the output of an FGS encoder. It is likely that, at the end of each fgs vop before the next fgs_vop_start_code, only partial bits of the fgs vop are at the input of the decoder due to truncation of the fgs vop bitstream. Decoding of the truncated bitstream is not normative. An example of dealing with the truncated bitstream is described in Annex S. The FGS syntax description in this clause is for a complete bitstream without truncation.

"

6) Replace Table 6-3 in subclause 6.2.1 with the following table:

"

Table 6-3. Start code values

name	**start code value (hexadecimal)**
video_object_start_code	00 through 1F
video_object_layer_start_code	20 through 2F
reserved	30 through 3F
fgs_bp_start_code	40 through 5F
reserved	60 through AF
visual_object_sequence_start_code	B0
visual_object_sequence_end_code	B1
user_data_start_code	B2
group_of_vop_start_code	B3
video_session_error_code	B4
visual_object_start_code	B5
vop_start_code	B6

slice_start_code	B7
extension_start_code	B8
fgs_vop_start_code	B9
fba_object_start_code	BA
fba_object_plane_start_code	BB
mesh_object_start_code	BC
mesh_object_plane_start_code	BD
still_texture_object_start_code	BE
texture_spatial_layer_start_code	BF
texture_snr_layer_start_code	C0
texture_tile_start_code	C1
texture_shape_layer_start_code	C2
reserved	C3-C5
System start codes (see note)	C6 through FF
NOTE System start codes are defined in ISO/IEC 14496-1	

"

7) Replace VideoObjectLayer() in subclause 6.2.3:

"

VideoObjectLayer() {	**No. of bits**	**Mnemonic**
if(next_bits() == video_object_layer_start_code) {		
short_video_header = 0		
video_object_layer_start_code	32	bslbf
random_accessible_vol	1	bslbf
video_object_type_indication	8	uimsbf
is_object_layer_identifier	1	uimsbf
if (is_object_layer_identifier) {		
video_object_layer_verid	4	uimsbf
video_object_layer_priority	3	uimsbf
}		
aspect_ratio_info	4	uimsbf
if (aspect_ratio_info == "extended_PAR") {		
par_width	8	uimsbf
par_height	8	uimsbf
}		
vol_control_parameters	1	bslbf
if (vol_control_parameters) {		
chroma_format	2	uimsbf
low_delay	1	uimsbf
vbv_parameters	1	blsbf
if (vbv_parameters) {		
first_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
latter_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
first_half_vbv_buffer_size	15	uimsbf

marker_bit	1	bslbf
latter_half_vbv_buffer_size	3	uimsbf
first_half_vbv_occupancy	11	uimsbf
marker_bit	1	blsbf
latter_half_vbv_occupancy	15	uimsbf
marker_bit	1	blsbf
}		
}		
video_object_layer_shape	2	uimsbf
if (video_object_layer_shape == "grayscale" && video_object_layer_verid != ‘0001’)		
video_object_layer_shape_extension	4	uimsbf
marker_bit	1	bslbf
vop_time_increment_resolution	16	uimsbf
marker_bit	1	bslbf
fixed_vop_rate	1	bslbf
if (fixed_vop_rate)		
fixed_vop_time_increment	1-16	uimsbf
if (video_object_layer_shape != “binary only”) {		
if (video_object_layer_shape == “rectangular”) {		
marker_bit	1	bslbf
video_object_layer_width	13	uimsbf
marker_bit	1	bslbf
video_object_layer_height	13	uimsbf
marker_bit	1	bslbf
}		
interlaced	1	bslbf
obmc_disable	1	bslbf
if (video_object_layer_verid == ‘0001’)		
sprite_enable	1	bslbf
else		
sprite_enable	2	uimsbf
if (sprite_enable== “static” || sprite_enable == “GMC”) {		
if (sprite_enable != “GMC”) {		
sprite_width	13	uimsbf
marker_bit	1	bslbf
sprite_height	13	uimsbf
marker_bit	1	bslbf
sprite_left_coordinate	13	simsbf
marker_bit	1	bslbf
sprite_top_coordinate	13	simsbf
marker_bit	1	bslbf
}		
no_of_sprite_warping_points	6	uimsbf
sprite_warping_accuracy	2	uimsbf
sprite_brightness_change	1	bslbf

if (sprite_enable != “GMC”)		
low_latency_sprite_enable	1	bslbf
}		
if (video_object_layer_verid != ‘0001’ && video_object_layer_shape != ”rectangular”)		
sadct_disable	1	bslbf
not_8_bit	1	bslbf
if (not_8_ bit) {		
quant_precision	4	uimsbf
bits_per_pixel	4	uimsbf
}		
if (video_object_layer_shape==”grayscale”) {		
no_gray_quant_update	1	bslbf
composition_method	1	bslbf
linear_composition	1	bslbf
}		
quant_type	1	bslbf
if (quant_type) {		
load_intra_quant_mat	1	bslbf
if (load_intra_quant_mat)		
intra_quant_mat	8*[2-64]	uimsbf
load_nonintra_quant_mat	1	bslbf
if (load_nonintra_quant_mat)		
nonintra_quant_mat	8*[2-64]	uimsbf
if(video_object_layer_shape==”grayscale”) {		
for(i=0; i<aux_comp_count; i++) {		
load_intra_quant_mat_grayscale	1	bslbf
if(load_intra_quant_mat_grayscale)		
intra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
load_nonintra_quant_mat_grayscale	1	bslbf
if(load_nonintra_quant_mat_grayscale)		
nonintra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
}		
}		
}		
if (video_object_layer_verid != ‘0001’)		
quarter_sample	1	bslbf
complexity_estimation_disable	1	bslbf
if (!complexity_estimation_disable)		
define_vop_complexity_estimation_header()		
resync_marker_disable	1	bslbf
data_partitioned	1	bslbf
if(data_partitioned)		
reversible_vlc	1	bslbf
if(video_object_layer_verid != ’0001’) {		

newpred_enable	1	bslbf
if (newpred_enable) {		
requested_upstream_message_type	2	uimsbf
newpred_segment_type	1	bslbf
}		
reduced_resolution_vop_enable	1	bslbf
}		
scalability	1	bslbf
if (scalability) {		
hierarchy_type	1	bslbf
ref_layer_id	4	uimsbf
ref_layer_sampling_direc	1	bslbf
hor_sampling_factor_n	5	uimsbf
hor_sampling_factor_m	5	uimsbf
vert_sampling_factor_n	5	uimsbf
vert_sampling_factor_m	5	uimsbf
enhancement_type	1	bslbf
if(video_object_layer == "binary" && hierarchy_type== '0') {		
use_ref_shape	1	bslbf
use_ref_texture	1	bslbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
}		
else {		
if(video_object_layer_verid !="0001") {		
scalability	1	bslbf
if(scalability) {		
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
resync_marker_disable	1	bslbf
}		
next_start_code()		
while (next_bits()== user_data_start_code){		
user_data()		
}		
if (sprite_enable == "static" && !low_latency_sprite_enable)		
VideoObjectPlane()		

do {		
if (next_bits() == group_of_vop_start_code)		
Group_of_VideoObjectPlane()		
VideoObjectPlane()		
} while ((next_bits() == group_of_vop_start_code) || (next_bits() == vop_start_code))		
} else {		
short_video_header = 1		
do {		
video_plane_with_short_header()		
} while(next_bits() == short_video_start_marker)		
}		
}		

"

with

"

VideoObjectLayer() {	**No. of bits**	**Mnemonic**
if(next_bits() == video_object_layer_start_code) {		
short_video_header = 0		
video_object_layer_start_code	32	bslbf
random_accessible_vol	1	bslbf
video_object_type_indication	8	uimsbf
if (video_object_type_indication == "Fine Granularity Scalable") {		
fgs_layer_type	2	uimsbf
video_object_layer_priority	3	uimsbf
aspect_ratio_info	4	uimsbf
if (aspect_ratio_info == "extended_PAR") {		
par_width	8	uimsbf
par_height	8	uimsbf
}		
vol_control_parameters	1	bslbf
if (vol_control_parameters) {		
chroma_format	2	uimsbf
low_delay	1	uimsbf
}		
marker_bit	1	bslbf
vop_time_increment_resolution	16	uimsbf
marker_bit	1	bslbf
fixed_vop_rate	1	bslbf
if (fixed_vop_rate)		
fixed_vop_time_increment	1-16	uimsbf
marker_bit	1	bslbf

video_object_layer_width	13	uimsbf
marker_bit	1	bslbf
video_object_layer_height	13	uimsbf
marker_bit	1	bslbf
interlaced	1	bslbf
if (fgs_layer_type ==“FGST” || fgs_layer_type ==“FGS_FGST”)		
fgs_ref_layer_id	4	uimsbf
if (fgs_layer_type ==“FGS” || fgs_layer_type ==“FGS_FGST”) {		
fgs_frequency_weighting_enable	1	bslbf
if (fgs_frequency_weighting_enable) {		
load_fgs_frequency_weighting_matrix	1	bslbf
if (load_fgs_frequency_weighting_matrix)		
fgs_frequency_weighting_matrix	3*[2-64]	uimsbf
}		
}		
if (fgs_layer_type ==“FGST” || fgs_layer_type ==“FGS_FGST”) {		
fgst_frequency_weighting_enable	1	bslbf
if (fgst_frequency_weighting_enable) {		
load_fgst_frequency_weighting_matrix	1	bslbf
if (load_fgst_frequency_weighting_matrix)		
fgst_frequency_weighting_matrix	3*[2-64]	uimsbf
}		
}		
quarter_sample	1	bslbf
fgs_resync_marker_disable	1	bslbf
do {		
if (nextbits_bytealigned() == group_of_vop_start_code)		
Group_of_VideoObjectPlane()		
FGSVideoObjectPlane()		
} while((nextbits_bytealigned()==group_of_vop_start_code)|| (nextbits_bytealigned()==fgs_vop_start_code))		
} else {		
is_object_layer_identifier	1	uimsbf
if (is_object_layer_identifier) {		
video_object_layer_verid	4	uimsbf
video_object_layer_priority	3	uimsbf
}		
aspect_ratio_info	4	uimsbf
if (aspect_ratio_info == “extended_PAR”) {		
par_width	8	uimsbf
par_height	8	uimsbf
}		

vol_control_parameters	1	bslbf
if (vol_control_parameters) {		
chroma_format	2	uimsbf
low_delay	1	uimsbf
vbv_parameters	1	blsbf
if (vbv_parameters) {		
first_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
latter_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
first_half_vbv_buffer_size	15	uimsbf
marker_bit	1	bslbf
latter_half_vbv_buffer_size	3	uimsbf
first_half_vbv_occupancy	11	uimsbf
marker_bit	1	blsbf
latter_half_vbv_occupancy	15	uimsbf
marker_bit	1	blsbf
}		
}		
video_object_layer_shape	2	uimsbf

if (video_object_layer_shape == "grayscale"
    && video_object_layer_verid != ‘0001’)		

video_object_layer_shape_extension	4	uimsbf
marker_bit	1	bslbf
vop_time_increment_resolution	16	uimsbf
marker_bit	1	bslbf
fixed_vop_rate	1	bslbf
if (fixed_vop_rate)		
fixed_vop_time_increment	1-16	uimsbf
if (video_object_layer_shape != “binary only”) {		
if (video_object_layer_shape == “rectangular”) {		
marker_bit	1	bslbf
video_object_layer_width	13	uimsbf
marker_bit	1	bslbf
video_object_layer_height	13	uimsbf
marker_bit	1	bslbf
}		
interlaced	1	bslbf
obmc_disable	1	bslbf
if (video_object_layer_verid == ‘0001’)		
sprite_enable	1	bslbf
else		
sprite_enable	2	uimsbf
if (sprite_enable== “static” || sprite_enable == “GMC”) {		
if (sprite_enable != “GMC”) {		
sprite_width	13	uimsbf
marker_bit	1	bslbf

sprite_height	13	uimsbf
marker_bit	1	bslbf
sprite_left_coordinate	13	simsbf
marker_bit	1	bslbf
sprite_top_coordinate	13	simsbf
marker_bit	1	bslbf
}		
no_of_sprite_warping_points	6	uimsbf
sprite_warping_accuracy	2	uimsbf
sprite_brightness_change	1	bslbf
if (sprite_enable != "GMC")		
low_latency_sprite_enable	1	bslbf
}		
if (video_object_layer_verid != '0001' && video_object_layer_shape != "rectangular")		
sadct_disable	1	bslbf
not_8_bit	1	bslbf
if (not_8_ bit) {		
quant_precision	4	uimsbf
bits_per_pixel	4	uimsbf
}		
if (video_object_layer_shape=="grayscale") {		
no_gray_quant_update	1	bslbf
composition_method	1	bslbf
linear_composition	1	bslbf
}		
quant_type	1	bslbf
if (quant_type) {		
load_intra_quant_mat	1	bslbf
if (load_intra_quant_mat)		
intra_quant_mat	8*[2-64]	uimsbf
load_nonintra_quant_mat	1	bslbf
if (load_nonintra_quant_mat)		
nonintra_quant_mat	8*[2-64]	uimsbf
if(video_object_layer_shape=="grayscale") {		
for(i=0; i<aux_comp_count; i++) {		
load_intra_quant_mat_grayscale	1	bslbf
if(load_intra_quant_mat_grayscale)		
intra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
load_nonintra_quant_mat_grayscale	1	bslbf
if(load_nonintra_quant_mat_grayscale)		
nonintra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
}		
}		
}		

if (video_object_layer_verid != ‘0001’)		
quarter_sample	1	bslbf
complexity_estimation_disable	1	bslbf
if (!complexity_estimation_disable)		
define_vop_complexity_estimation_header()		
resync_marker_disable	1	bslbf
data_partitioned	1	bslbf
if(data_partitioned)		
reversible_vlc	1	bslbf
if(video_object_layer_verid != ’0001’) {		
newpred_enable	1	bslbf
if (newpred_enable) {		
requested_upstream_message_type	2	uimsbf
newpred_segment_type	1	bslbf
}		
reduced_resolution_vop_enable	1	bslbf
}		
scalability	1	bslbf
if (scalability) {		
hierarchy_type	1	bslbf
ref_layer_id	4	uimsbf
ref_layer_sampling_direc	1	bslbf
hor_sampling_factor_n	5	uimsbf
hor_sampling_factor_m	5	uimsbf
vert_sampling_factor_n	5	uimsbf
vert_sampling_factor_m	5	uimsbf
enhancement_type	1	bslbf
if(video_object_layer == “binary” && hierarchy_type== ‘0’) {		
use_ref_shape	1	bslbf
use_ref_texture	1	bslbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
} else {		
if(video_object_layer_verid !=”0001”) {		
scalability	1	bslbf
if(scalability) {		
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		

resync_marker_disable	1	bslbf
}		
next_start_code()		
while (next_bits()== user_data_start_code) {		
user_data()		
}		
if (sprite_enable == “static” && !low_latency_sprite_enable)		
VideoObjectPlane()		
do {		
if (next_bits() == group_of_vop_start_code)		
Group_of_VideoObjectPlane()		
VideoObjectPlane()		
} while ((next_bits() == group_of_vop_start_code) || (next_bits() == vop_start_code))		
}		
} else {		
short_video_header = 1		
do {		
video_plane_with_short_header()		
} while(next_bits() == short_video_start_marker)		
}		
}		

“

8) Add the following subclause 6.2.14 after subclause 6.2.13:

“

6.2.14 FGS Video Object

6.2.14.1 FGS Video Object Plane

FGSVideoObjectPlane() {	**No. of bits**	**Mnemonic**
fgs_vop_start_code	32	bslbf
fgs_vop_coding_type	2	uimsbf
do {		
modulo_time_base	1	bslbf
} while (modulo_time_base != ‘0’)		
marker_bit	1	bslbf
vop_time_increment	1-16	uimsbf
marker_bit	1	bslbf

fgs_vop_max_level_y	5	uimsbf
fgs_vop_max_level_u	5	uimsbf
fgs_vop_max_level_v	5	uimsbf
marker_bit	1	bslbf
fgs_vop_number_of_vop_bp_coded	5	uimsbf
fgs_vop_mc_bit_plane_used	5	uimsbf
fgs_vop_selective_enhancement_enable	1	bslbf
if (fgs_vop_coding_type != "I") {		
if (interlaced)		
top_field_first	1	bslbf
vop_fcode_forward	3	uimsbf
if (fgs_vop_coding_type == "B")		
vop_fcode_backward	3	uimsbf
fgs_ref_select_code	2	uimsbf
do {		
fgs_motion_macroblock()		
} while(nextbits_bytealigned() != '000 0000 0000 0000 0000 0000')		
}		
next_start_code()		
if (nextbits_bytealigned () == fgs_bp_start_code) {		
while(nextbits_bytealigned() != '000 0000 0000 0000 0000 0000' || nextbits_bytealigned () == fgs_bp_start_code) {		
if (start_of_bit_plane())		
fgs_bp_start_code	32	bslbf
else {		
if (! fgs_resync_marker_disable && nextbits_bytealigned () == fgs_resync_marker) {		
next_resync_marker()		
fgs_resync_marker	23	uimsbf
fgs_ macroblock_number	1-14	vlclbf
}		
}		
fgs_macroblock()		
}		
next_start_code()		
}		
}		

6.2.14.2 FGS Motion Macroblock

fgs_motion_macroblock() {	**No. of bits**	**Mnemonic**
if (fgs_vop_coding_type == “P”) {		
fgs_not_coded	1	bslbf
if (!fgs_not_coded) {		
fgs_p_mb_type	1	bslbf
if (interlaced)		
fgs_motion_interlaced_information()		
if (fgs_p_mb_type == 0) {		
fgs_motion_vector(“forward”)		
if (interlaced && field_prediction)		
fgs_motion_vector(“forward”)		
}		
if (fgs_p_mb_type == 1) {		
for (j=0; j < 4; j++)		
fgs_motion_vector(“forward”)		
}		
}		
} else {		
fgs_modb	1	bslbf
if (!fgs_modb) {		
fgs_b_mb_type	1-4	vlclbf
if (interlaced)		
fgs_motion_interlaced_information()		
if (fgs_b_mb_type==‘0001’ || fgs_b_mb_type==‘01’) {		
fgs_motion_vector(“forward”)		
if (interlaced && field_prediction)		
fgs_motion_vector(“forward”)		
}		
if (fgs_b_mb_type == ‘01’ || fgs_b_mb_type == ‘001’) {		
fgs_motion_vector(“backward”)		
if (interlaced && field_prediction)		
fgs_motion_vector(“backward”)		

}		
if (fgs_b_mb_type == '1')		
fgs_motion_vector("direct")		
}		
}		
}		

6.2.14.3 FGS Motion Interlaced Information

fgs_motion_interlaced_information( ) {	No. of bits	Mnemonic
fgs_field_prediction	1	bslbf
if (fgs_field_prediction) {		
if (fgs_vop_coding_type == "P" || (fgs_vop_coding_type=="B" && fgs_b_mb_type!="001")){		
fgs_forward_top_field_reference	1	bslbf
fgs_forward_bottom_field_reference	1	bslbf
}		
if ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1")) {		
fgs_backward_top_field_reference	1	bslbf
fgs_backward_bottom_field_reference	1	bslbf
}		
}		
}		

6.2.14.4 FGS Motion Vector

fgs_motion_vector (mode) {	No. of bits	Mnemonic
if (mode == „direct“) {		
horizontal_mv_data	1-13	vlclbf
vertical_mv_data	1-13	vlclbf
}		
else if (mode == „forward“) {		
horizontal_mv_data	1-13	vlclbf

if ((vop_fcode_forward != 1)&&(horizontal_mv_data != 0))		
horizontal_mv_residual	1-6	uimsbf
vertical_mv_data	1-13	vlclbf
if ((vop_fcode_forward != 1)&&(vertical_mv_data != 0))		
vertical_mv_residual	1-6	uimsbf
}		
else if (mode == „backward“) {		
horizontal_mv_data	1-13	vlclbf
if ((vop_fcode_backward != 1)&&(horizontal_mv_data != 0))		
horizontal_mv_residual	1-6	uimsbf
vertical_mv_data	1-13	vlclbf
if ((vop_fcode_backward != 1)&&(vertical_mv_data != 0))		
vertical_mv_residual	1-6	uimsbf
}		
}		

6.2.14.5 FGS Macroblock

fgs_macroblock() {	**No. of bits**	**Mnemonic**
if (fgs_vop_bp_id < 2)		
fgs_cbp	1-9	vlclbf
if (fgs_vop_selective_enhancement_enable==1) {		
if (!mb_shift_factor_received && none_zero_macroblock)		
fgs_shifted_bits	1-5	vlclbf
}		
if (interlaced==1) {		
if (!dct_type_received && non_zero_macroblock)		
fgs_dct_type	1	bslbf
}		
for (i=0; i<6; i++) {		
if (start_decode==1)		
fgs_block()		
}		
}		

NOTE 1 — In the syntax of fgs vop, there are three elements: fgs_vop_max_level_y, fgs_vop_max_level_u, and fgs_vop_max_level_v. The maximum value of these three elements is the number of vop-bps in an fgs vop. If any one of the three elements has a smaller value than the number of vop-bps in the fgs vop, the color component corresponding to the smaller value element is absent in one or more vop-bps. If the difference between the number of vop-bps and the value of the element is k, the corresponding color component is absent in the first k vop-bps. There is no need to decode fgs blocks of the color component absent in the vop-bps. start_decode is defined to be a flag to indicate whether decoding should be performed for an fgs block depending on whether the fgs block belongs to an absent color component or not. The value of start_decode is 0 if the fgs block belongs to an absent color component. Otherwise, the value of start_decode is 1 and decoding is performed for the fgs block.

NOTE 2 — mb_shift_factor_received is a flag to indicate whether fgs_shifted_bits has been decoded in the previous fgs macroblock with the same fgs macroblock number. It is initialized to 0 before decoding the first vop-bp. It is set to 1 after fgs_shifted_bits is decoded.

NOTE 3 — non_zero_macroblock is a flag to indicate whether every block-bp in the fgs macroblock is an all-zero block-bp. If every block-bp in the fgs macroblock is an all-zero block-bp, the value of this flag is 0. Otherwise, its value is 1.

NOTE 4 — dct_type_received is a flag to indicate whether fgs_dct_type has been decoded in the previous fgs macroblock with the same fgs macroblock number. It is initialized to 0 before decoding the first vop-bp. It is set to 1 after fgs_dct_type is decoded.

6.2.14.6 FGS Block

fgs_block() {	**No. of bits**	**Mnemonic**
if (fgs_vop_bp_id>=2 && previous_fgs_msb_not_reached==1)		
fgs_msb_not_reached	1	bslbf
if (fgs_msb_not_reached == 0) {		
while (eop == 0) {		
fgs_run_eop_code	1-16	vlclbf
if (coeff_msb_not_reached ==1)		
fgs_sign_bit	1	bslbf
}		
}		
}		

NOTE 1 — previous_fgs_msb_not_reached is defined to be fgs_msb_not_reached decoded in the previous block-bp of the same 8x8 DCT block.

NOTE 2 — eop is defined to be the EOP flag resulting from decoding the most recent fgs_run_eop_code. eop is reset to 0 at the beginning of fgs_block().

NOTE 3 — coeff_msb_not_reached is defined to be an internal flag indicating, with a value '1', that the MSB of the non-zero DCT coefficient associated with the fgs_run_eop_code above was not reached. The value of this flag is changed to '0' when the MSB of the non-zero DCT coefficient is reached.

"

9) Replace Table 6-4 in subclause 6.3.2 with the following table:

"

Table 6-4 — Meaning of visual_object_verid

Visual_object_verid	**Meaning**
0000	reserved
0001	object type listed in Table 9-1
0010	object type listed in Table V2-39
0011	reserved
0100	object type listed in Table AMD1-40
0101	object type listed in Table AMD2-13
0110 - 1111	reserved

"

10) Replace Table 6-10 in subclause 6.3.3 with the following table:

"

Table 6-10 — FLC table for video_object_type_indication

Video Object Type	**Code**
Reserved	00000000
Simple Object Type	00000001
Simple Scalable Object Type	00000010
Core Object Type	00000011
Main Object Type	00000100
N-bit Object Type	00000101
Basic Anim. 2D Texture	00000110
Anim. 2D Mesh	00000111
Simple Face	00001000
Still Scalable Texture	00001001
Advanced Real Time Simple	00001010
Core Scalable	00001011
Advanced Coding Efficiency	00001100
Advanced Scalable Texture	00001101
Simple FBA	00001110
Simple Studio	00001111
Core Studio	00010000
Advanced Simple	00010001
Fine Granularity Scalable	00010010
Reserved	00010011 - 11111111

"

11) Add the following to subclause 6.3.3 after Table 6-10:

"

fgs_layer_type – This is a 2-bit code indicating whether this layer is FGS only, FGST only, or a combination of FGS and FGST. Table AMD2-1 shows the codes and the meanings.

Table AMD2-1 — Code for fgs_layer_type

Code	Meaning
00	reserved
01	FGS
10	FGST
11	FGS-FGST

"

12) Replace Table 6-11 in subclause 6.3.3 with the following table:

"

Table 6-11 – Meaning of video_object_layer_verid

video_object_layer_verid	Meaning
0000	reserved
0001	object type listed in Table 9-1
0010	object type listed in Table V2-39
0011	reserved
0100	object type listed in Table AMD1-40
0101	object type listed in Table AMD2-13
0011 - 1111	reserved

"

13) Replace the following in subclause 6.3.3:

"

video_object_layer_priority – This is a 3-bit code which specifies the priority of the video object layer. It takes values between 1 and 7, with 1 representing the highest priority and 7, the lowest priority. The value of zero is reserved.

"

with

"

video_object_layer_priority – This is a 3-bit code which specifies the priority of the video object layer. It takes values between 1 and 7, with 1 representing the highest priority and 7 the lowest priority. The value of zero is reserved. For the transmission of FGS and FGST in two VOLs, the relative transmission priority of an FGS VOL vs.

that of an FGST VOL can be specified by setting this parameter in the FGS VOL relative to the same parameter in the FGST VOL.

"

14) Add the following to subclause 6.3.3 after 'Interlaced':

"

fgs_ref_layer_id – This is a 4-bit unsigned integer with value between 0 and 15. It indicates the layer to be used as reference for prediction in the case of fgs_layer_type being FGST or FGS_FGST.

fgs_frequency_weighting_enable – This is a one-bit flag to indicate that frequency weighting is used in this VOL, when set to '1'. Otherwise, when this flag is set to '0', frequency weighting is not used. The default frequency weighting matrix is an all zero matrix when fgs_frequency_weighting_enable is '1'.

load_fgs_frequency_weighting_matrix – This is a one-bit flag which is set to '1' when fgs_frequency_weighting_matrix follows. If it is set to '0' then the default frequency weighting matrix is used.

fgs_frequency_weighting_matrix – This is a list of 2 to 64 three-bit unsigned integers. The integers are in zigzag scan order representing the fgs_frequency_weighting_matrix. A value of 0 indicates that no more values are transmitted and the remaining, non-transmitted values are set to zero.

fgst_frequency_weighting_enable – This is a one-bit flag to indicate that frequency weighting is used in this VOL, when set to '1'. Otherwise, when this flag is set to '0', frequency weighting is not used. The default frequency weighting matrix is an all zero matrix when fgst_frequency_weighting_enable is '1'.

load_fgst_frequency_weighting_matrix – This is a one-bit flag which is set to '1' when fgst_frequency_weighting_matrix follows. If it is set to '0' then the default matrix is used.

fgst_frequency_weighting_matrix – This is a list of 2 to 64 three-bit unsigned integers. The integers are in zigzag scan order representing the fgst_frequency_weighting_matrix. A value of 0 indicates that no more values are transmitted and the remaining, non-transmitted values are set to zero.

fgs_resync_marker_disable – This is a one-bit flag which when set to '1' indicates that there is no fgs_resync_marker in coded fgs vops of this VOL. When this flag is set to '0', it indicates that fgs_resync_marker may be used in coded fgs vops of this VOL.

"

15) Replace the following in subclause 6.3.3:

"

sprite_enable: When video_object_layer_verid == '0001', this is a one-bit flag which when set to '1' indicates the usage of static (basic or low latency) sprite coding. When video_object_layer_verid == '0002', this is a two-bit unsigned integer which indicates the usage of static sprite coding or global motion compensation (GMC). Table V2-2 shows the meaning of various codewords. An S-VOP with sprite_enable == "GMC" is referred to as an S (GMC)-VOP in this document.

Table V2 - 2 – Meaning of sprite_enable codewords

sprite_enable (video_object_layer_ verid == ‘0001’)	sprite_enable (video_object_layer_ verid == ‘0002’)	Sprite Coding Mode
0	00	sprite not used
1	01	static (Basic/Low Latency)
–	10	GMC (Global Motion Compensation)
–	11	reserved

"

with

"

sprite_enable: When video_object_layer_verid == ‘0001’, this is a one-bit flag which when set to ‘1’ indicates the usage of static (basic or low latency) sprite coding. When video_object_layer_verid == ‘0010’ or video_object_layer_verid == ‘0101’, this is a two-bit unsigned integer which indicates the usage of static sprite coding or global motion compensation (GMC). Table V2-2 shows the meaning of various codewords. An S-VOP with sprite_enable == “GMC” is referred to as an S (GMC)-VOP in this document.

Table V2 - 2 – Meaning of sprite_enable codewords

sprite_enable (video_object_layer_ verid == ‘0001’)	sprite_enable (video_object_layer_verid == '0010' || video_object_layer_verid == '0101')	Sprite Coding Mode
0	00	sprite not used
1	01	static (Basic/Low Latency)
–	10	GMC (Global Motion Compensation)
–	11	reserved

"

16) Replace the following in subclause 6.3.3:

"

quarter_sample: This is a one-bit flag which when set to ‘0’ indicates that half sample mode and when set to ‘1’ indicates that quarter sample mode shall be used for motion compensation of the luminance component.

"

with

"

quarter_sample: This is a one-bit flag which when set to ‘0’ indicates that half sample mode and when set to ‘1’ indicates that quarter sample mode shall be used for motion compensation of the luminance component. For FGST or FGS_FGST enhancement layer, this flag shall be 0.

"

17) Add the following subclause 6.3.14 after subclause 6.3.13:

"

6.3.14 FGS Video Object

6.3.14.1 FGS Video Object Plane

fgs_vop_start_code – This is the bit string '000001B9' in hexadecimal. It marks the start of an fgs vop.

fgs_vop_coding_type – The fgs_vop_coding_type identifies whether an fgs vop is an FGS coding type (I), predictive-coded FGS coding type (P), or bidirectionally predictive-coded FGS coding type (B). The meaning of fgs_vop_coding_type is defined in Table AMD2-2.

Table AMD2-2 — Meaning of fgs_vop_coding_type

fgs_vop_coding_type	coding method
00	FGS (I)
01	predictive-coded FGS (P)
10	bidirectionally-predictive-coded FGS (B)
11	reserved

fgs_vop_max_level_y – This is an unsigned integer which specifies the number of vop-bps in which the Y component has at least one non-zero block-bp in the first one of them.

fgs_vop_max_level_u – This is an unsigned integer which specifies the number of vop-bps in which the U component has at least one non-zero block-bp in the first one of them.

fgs_vop_max_level_v – This is an unsigned integer which specifies the number of vop-bps in which the V component has at least one non-zero block-bp in the first one of them.

fgs_vop_number_of_vop_bp_coded – This is an unsigned integer which specifies the number of vop-bps coded into the bitstream.

fgs_vop_mc_bit_plane_used – This parameter is reserved for future use. The value of this parameter shall be 0.

fgs_vop_selective_enhancement_enable – This flag shall be 1 when selective enhancement is enabled and it shall be 0 otherwise.

fgs_ref_select_code – This is a 2-bit unsigned integer which specifies prediction reference choices for "P" and "B" fgs vops with respect to decoded reference layer identified by ref_layer_id. The meaning of allowed values is specified in Table AMD2-3 and Table AMD2-4.

Table AMD2-3 — Prediction reference choices for "P" fgs vops

fgs_ref_select_code	forward prediction reference
00	Reserved
01	Most recently VOP in display order belonging to the reference layer.
10	Next VOP in display order belonging to the reference layer.
11	Reserved

Table AMD2-4 — Prediction reference choices for "B" fgs vops

fgs_ref_select_code	forward temporal reference	backward temporal reference
00	Reserved	
01	Reserved	
10	Reserved	
11	Most recently VOP in display order belonging to the reference layer.	Next VOP in display order belonging to the reference layer.

fgs_bp_start_code – This is a string of 32 bits. The first 27 bits are '0000 0000 0000 0000 0000 0001 010' in binary and the last 5 bits represent one of the values in the range of '00000' to '11111' in binary. The fgs_bp_start_code marks a new vop-bp.

fgs_vop_bp_id – This is given by the last 5 bits of the fgs_bp_start_code. The fgs_vop_bp_id uniquely identifies a vop-bp. The value of fgs_vop_bp_id is "0" for the most significant vop-bp and increments by 1 for each successively lower vop-bp.

fgs_resync_marker – This is a binary string of 22 zeros followed by a one '000 0000 0000 0000 0000 0001'. It is only present when fgs_resync_marker_disable flag is set to '0'.

fgs_macroblock_number – This is a variable length code with length between 1 and 14 bits. It identifies the fgs macroblock number of the following fgs macroblock. The actual length of the code depends on the total number of fgs macroblocks in a vop-bp, calculated according to Table AMD2-5. The code itself is a binary representation of the fgs macroblock number.

Table AMD2-5 – Length of fgs_macroblock_number code

length of fgs_macroblock_number code	((video_object_layer_width+15)/16) * ((video_object_layer_height+15)/16)
1	1-2
2	3-4
3	5-8
4	9-16
5	17-32
6	33-64
7	65-128
8	129-256
9	257-512
10	513-1024

11	1025-2048
12	2049-4096
13	4097-8192
14	8193-16384

6.3.14.2 FGS Motion Macroblock

fgs_not_coded – This is a 1-bit flag that signals if an FGS motion macroblock is coded or not in a “P” fgs vop. When set to’1’ it indicates that an FGS motion macroblock is not coded and no further data is included in the bitstream for this FGS motion macroblock; decoder shall treat this FGS motion macroblock as motion vector equal to zero. When set to ‘0’ it indicates that the FGS motion macroblock is coded and its data is included in the bitstream.

fgs_p_mb_type – This is a 1-bit flag that signals the type of motion vector coding in “P” fgs vop. When set to “0”, it indicates that the one-motion-vector mode is used in this FGS motion macroblock. When set to “1”, it indicates that the four-motion-vector mode is used in this FGS motion macroblock.

fgs_modb – This is a 1-bit flag that signals if an FGS motion macroblock is coded or not in a “B” fgs vop. When set to’1’ it indicates that an FGS motion macroblock is not coded and no further data is included in the bitstream for this FGS motion macroblock; decoder shall treat this FGS motion macroblock as motion vector equal to zero. When set to ‘0’ it indicates that the FGS motion macroblock is coded and its data is included in the bitstream.

fgs_b_mb_type – This variable length code is present only in coded FGS motion macroblocks of “B” fgs vops. The codes for fs_b_mb_type are shown in Table AMD2-6.

Table AMD2-6 — VLC table for fgs_b_mb_type

Code	fgs_b_mb_type
1	direct
01	interpolate
001	backward
0001	forward

6.3.14.3 FGS Motion Interlaced Information

fgs_field_prediction – This is a 1-bit flag indicating whether the FGS motion macroblock is field predicted or frame predicted. This flag is set to ‘1’ when the FGS motion macroblock is predicted using field motion vectors. If it is set to ‘0’ then frame prediction (16x16 or 8x8) will be used. This flag is only present in the bitstream if the interlaced flag is set to “1”, and either the fgs_p_mb_type is “0” in a “P” fgs vop or the FGS motion macroblock is in a “B” fgs vop.

fgs_forward_top_field_reference – This is a 1-bit flag which indicates the reference field for the forward motion compensation of the top field. When this flag is set to ‘0’, the top field is used as the reference field. If it is set to ‘1’ then the bottom field will be used as the reference field. This flag is only present in the bitstream if the fgs_field_prediction flag is set to “1” and the FGS motion macroblock is not backward predicted.

fgs_forward_bottom_field_reference – This is a 1-bit flag which indicates the reference field for the forward motion compensation of the bottom field. When this flag is set to ‘0’, the top field is used as the reference field. If it is set to ‘1’ then the bottom field will be used as the reference field. This flag is only present in the bitstream if the fgs_field_prediction flag is set to “1” and the FGS motion macroblock is not backward predicted.

fgs_backward_top_field_reference – This is a 1-bit flag which indicates the reference field for the backward motion compensation of the top field. When this flag is set to ‘0’, the top field is used as the reference field. If it is set to ‘1’ then the bottom field will be used as the reference field. This flag is only present in the bitstream if the fgs_field_prediction flag is set to “1” and the FGS motion macroblock is not forward predicted.

fgs_backward_bottom_field_reference – This is a 1-bit flag which indicates the reference field for the backward motion compensation of the bottom field. When this flag is set to ‘0’, the top field is used as the reference field. If it is set to ‘1’ then the bottom field will be used as the reference field. This flag is only present in the bitstream if the fgs_field_prediction flag is set to “1” and the FGS motion macroblock is not forward predicted.

6.3.14.4 FGS Motion Vector

Identical to subclause 6.3.6.2.

6.3.14.5 FGS Macroblock

fgs_cbp – This is a variable length code with length between 1 and 9 bits. It specifies the coded bit pattern of fgs_msb_not_reached in the fgs macroblock in the first two vop-bps.

For the first vop-bp with Y, U, and V components, Table AMD2-7 summarizes the code for fgs_cbp: (note: x denotes either 0 or 1)

Table AMD2-7 — Coding table for fgs_cbp in the first vop-bp with Y, U, and V components

Code	Pattern of fgs_msb_not_reached yyyy,uv	Meaning
1	1111,11	6 all-zero block-bps
01xxxx	xxxx,11	U and V block-bps are all-zero block-bps, but at least one of the 4 Y block-bps is not an all-zero block-bp
00xxxx01	xxxx,00	both U and V block-bps are not all-zero block-bps
00xxxx10	xxxx,01	U block-bp is not an all-zero block-bp and V block-bp is an all-zero block-bp
00xxxx11	xxxx,10	U block-bp is an all-zero block-bp and V block-bp is not an all-zero block-bp

Within a vop-bp, one of the three color components may be absent. If one of the U and V components is absent, the above coding scheme is followed except that the absent color component is not coded. Table AMD2-8 specifies this case:

Table AMD2-8 — Coding table for fgs_cbp in the first vop-bp when one of the U and V components is absent

Code	Pattern of fgs_msb_not_reached yyyy,u/v	Meaning
1	1111,1	5 all-zero block-bps
0xxxxx (000001 –011111)	xxxx,x (0000,0 – 1111,0)	not all 5 blocks are all-zero block-bps

To avoid start code emulation, the 5-bit pattern of fgs_msb_not_reached forms a 5-bit integer and the integer plus 1 in binary is 'xxxxx' in the code.

If both U and V components are absent in vop-bp, Table AMD2-9 specifies the code:

Table AMD2-9 — Coding table for fgs_cbp in the first vop-bp when both U and V components are absent

Code	Pattern of fgs_msb_not_reached yyyy	Meaning
1	1111	4 all-zero block-bps
0100 0101 0110 0111	0111 1011 1101 1110	3 all-zero block-bps
001000 001001 001010 001011 001100 001101	0011 0101 0110 1001 1010 1100	2 all-zero block-bps
000100 000101 000110 000111	1000 0100 0010 0001	1 all-zero block-bp
00001	0000	no all-zero block-bp

In the following unlikely cases:

- Y component is absent,
- the Y and U components are absent,
- the Y and V components are absent,

the pattern of fgs_msb_not_reached is put into the bitstream without VLC coding.

For the second vop-bp, the pattern of fgs_msb_not_reached is put into the bitstream without VLC coding except for the fgs macroblocks that satisfy the following conditions:

- Y, U, and V components of the second vop-bp are all present;
- The pattern of fgs_msb_not_reached for the fgs macroblock in the first vop-bp with the same fgs macroblock number is all 1's.

The coding method for this case is specified in Table AMD2-10:

Table AMD2-10 — Coding table for fgs_cbp in the second vop-bp

Code	Pattern of fgs_msb_not_reached yyyy,uv	Meaning
011	1111,11	6 all-zero block-bps
010	0000,11	4 Y block-bps are not all-zero block-bps and both U and V block-bps are all-zero block-bps
1xxxx	xxxx,11	at least one Y block-bp is all-zero block-bp but not all 4 Y block-bps are all-zero block-bps, and both U and V block-bps are all-zero block-bps
10000	0000,00	none of the 6 block-bps are all-zero block-bps
11111	1111,00	4 Y block-bps are all-zero block-bps and both U and V block-bps are not all-zero block-bps
00100	0000,01	4 Y block-bps and the U block-bp are not all-zero block-bps, but the V block-bp is an all-zero block-bp
00101	0000,10	4 Y block-bps and the V block-bp are not all-zero block-bps, but the U block-bp is an all-zero block-bp
00110	1111,01	4 Y block-bps and the V block-bp are all-zero block-bps, but the U block-bp is not an all-zero block-bp
00111	1111,10	4 Y block-bps and the U block-bp are all-zero block-bps, but the V block-bp is not an all-zero block-bp
000xxxxxx	xxxx,xx	All other cases

fgs_shifted_bits – This is a variable length code with length between 1 and 5 bits as shown in Table AMD2-11. When selective enhancement is used, all DCT coefficients in an fgs macroblock are left-shifted by fgs_shifted_bits.

Table AMD2-11 — VLC table of fgs_shifted_bits

Number of bits	VLC code
0	1
1	01
2	001
3	0001
4	00001

fgs_dct_type – This is a 1-bit flag indicating whether this fgs macroblock is frame DCT coded or field DCT coded. If this flag is set to "1", the fgs macroblock is field DCT coded; otherwise, the fgs macroblock is frame DCT coded. This flag is only present in the bitstream if the interlaced flag is set to "1" in the fgs vop header. This flag is only present once in the first non-zero fgs macroblock with the same fgs macroblock number.

6.3.14.6 FGS Block

fgs_msb_not_reached – This flag is 1 if the most significant block-bp in an 8x8 DCT block is not reached yet. In other words, this indicates an all-zero block-bp before the most significant block-bp. This flag is 0 otherwise.

fgs_run_eop_code – The VLC code for a (RUN,EOP) symbol as described in Clause 7.

fgs_sign_bit – This bit indicates the sign of the DCT coefficient with value 0 indicating positive and value 1 indicating negative.

"

18) Replace the following in subclause 7.6.9.5.2:

"

...where {(MVx[i],MVy[i]), i = 0,1,2,3} are the MV vectors of the co-located macroblock, TRD is the difference in temporal reference of the B-VOP and the previous reference VOP. TRD is the difference in temporal reference of the temporally next reference VOP with temporally previous reference VOP, assuming B-VOPs or skipped VOPs in between.

"

with

"

...where {(MVx[i],MVy[i]), i = 0,1,2,3} are the MV vectors of the co-located macroblock, TRD is the difference in temporal reference of the B-VOP and the previous reference VOP. TRD is the difference in temporal reference of the temporally next reference VOP with temporally previous reference VOP, assuming B-VOPs or skipped VOPs in between. In case that the MV components of the co-located macroblock are given in quarter sample units and the components MVDx and MVDy of the delta vector are given in half sample units, the components of the co-located macroblock {(MVx[i],MVy[i]), i = 0,1,2,3} are converted to half sample units before the calculation of the forward and the backward motion vectors {(MVFx[i], MVFy[i]), (MVBx[i], MVBy[i]), i = 0,1,2,3}. For this conversion, each component {(MVx[i],MVy[i]), i = 0,1,2,3} is first divided by 2 and then rounded on the basis of Table 7-9.

"

19) Add the following subclause 7.17 after subclause 7.16:

"

7.17 The FGS decoding process

This clause specifies the FGS decoding process that the decoder shall perform to recover visual data from the coded bitstream and reconstruct the enhancement VOP. In a typical application of FGS, the bitstream at the input of an FGS decoder is a truncated version of the bitstream at the output of an FGS encoder. It is likely that, at the end of each fgs vop bitstream before the next fgs_vop_start_code, only partial bits of an fgs block are at the input of the decoder due to truncation of the fgs vop bitstream. These partial bits of an fgs block may be discarded. The description of the FGS decoding process in the following sub-sections is for an fgs block that contains all of its bits.

7.17.1 Bit-plane decoding of the absolute values of the DCT coefficients

The bit-plane decoding procedure can be described as follows:

- The absolute values of all the DCT coefficients are initialized to zeros.

- Variable length decoding of (RUN, EOP) symbols is performed. If frequency weighting is used, i.e., **fgs_frequency_weighting_enable** is '1', the choice of VLC tables for decoding depends on the first value of the frequency weighting matrix (shifted bits for the DC component). Table AMD2-12 indicates this dependency:

Table AMD2-12 — VLC Table Choices for Frequency Weighting

fw_matrix[0]	VLC tables
0	B.4.1 table_bpc
1	B.4.1 table_bpc
2	B.4.2 table_bpc4fw_m
3	B.4.2 table_bpc4fw_m
4	B.4.3 table_bpc4fw_h
5	B.4.3 table_bpc4fw_h
6	B.4.3 table_bpc4fw_h
7	B.4.3 table_bpc4fw_h

where table_bpc, table_ bpc4fw_m, and table_bpc4fw_h are given in Annex B.4. Four 2-D VLC tables are used in each set. The first table in each set corresponds to the MSB block-bp. The second table in each set corresponds to the second MSB block-bp. The third table in each set corresponds to the third MSB block-bp. The fourth table in each set corresponds to the fourth MSB block-bp and all the lower block-bps. For the ESCAPE cases, RUN is represented with 6 bits to represent values from 0 to 63 and EOP is a 1-bit flag to indicate whether the end of the block-bp is reached (with value 1) or not (with value 0).

- The (RUN,EOP) symbols are translated into RUN number of consecutive 0's before a 1 and whether to fill 0's to the end of this block-bp. If the ALL-ZERO symbol is decoded, this block-bp contains all 0's. The ALL-ZERO symbol is not in the VLC tables for the MSB block-bp.

- Accumulate the bit to the partial result of each absolute value of the (bit shifted) DCT coefficients at the proper significant position.

- If frequency weighting or selective enhancement is used, the bit shifted DCT coefficient is right-shifted by S bits where S = fw_matrix[i]+ fgs_shifted_bits, and fw_matrix[i] is the i^{th} element of the frequency weighting matrix.

- After decoding all the bits for the fgs vop available to the FGS decoder, partial absolute values of all the DCT coefficients are available.

Note that the MSB block-bp in the above description is defined as the first block-bp of an 8x8 DCT block that contains at least one "1" bit value. The MSB block-bp of an 8x8 DCT block may or may not be in the first vop-bp.

7.17.2 Decoding of the signs of the DCT coefficients

A sign bit is decoded from the enhancement layer bitstream immediately after the (RUN, EOP) code corresponding to the MSB of the non-zero DCT coefficient.

7.17.3 Reconstruction of the enhancement DCT coefficients

Let DIFF and SIGN be the absolute value and sign of a DCT coefficient decoded from the enhancement bitstream, and ENH_COEFF be the reconstructed enhancement DCT coefficient.

ENH_COEFF = SIGN*DIFF

7.17.4 Reconstruction of the enhanced VOP

After enhanced DCT coefficients are obtained and inverse zigzag scan is used to convert the enhanced DCT coefficients into 8x8 blocks, an inverse DCT is applied to each 8x8 block. Inverse zigzag scan is performed according to Figure 7-4 (c) in subclause 7.4.2. If **fgs_vop_coding_type** is “I”, the reconstructed residue VOP is added to the base-layer VOP pixel by pixel. As a reference to this operation, the base-layer VOP is the one after clipping. If **fgs_vop_coding_type** is “P” or “B”, the reconstructed block after the inverse DCT followed by adding the motion compensated prediction block is the enhancement VOP. Motion vector decoding is performed according to subclause 7.6.3. Motion compensation is performed according to the applicable parts of subclause 7.9.1.1, subclause 7.9.1.3.2, and subclause 7.9.1.3.3. The reconstructed enhancement VOP pixels are limited into the value range between 0 and 255 by the clipping unit in the FGS enhancement layer to generate the final enhancement VOP.

“

20) Add the following Table in subclause 9.1 after tableV2-39 :

“

Table AMD2-13 — Video Object Types

Visual Tools	**Visual Object Types**	
	Advanced Simple	**Fine Granularity Scalable**
I-VOP	X	X
P-VOP	X	X
B-VOP	X	X
DC Prediction	X	X
AC Prediction	X	X
4-MV, Unrestricted MV	X	X
Slice Resynchronization	X	X
Data Partitioning	X	X
Reversible VLC	X	X
Short Header	X	X
Method 1/Method 2 Quantization	X	X
Interlace	X	X
Global Motion Compensation	X	
Quarter-pel Motion Compensation	X	
Fine Granularity Scalability		X
FGS Temporal Scalability		X

NOTE 1 — The interlace tools are not used for levels L0, L1, L2, and L3 of AS and FGS Profiles.

NOTE 2 — The Advanced Simple Object Type is the base-layer of the Fine Granularity Scalable Object Type. If FGS Temporal Scalability is used, B-VOP is not allowed in the base-layer. However, B-VOP is allowed in the base-layer when FGS Temporal Scalability is not used in the Fine Granularity Scalable Object Type.

NOTE 3 — The Advanced Simple Object Type is the base-layer of the Fine Granularity Scalable Object Type. If FGS or FGS Temporal Scalability is used, Short Header is not allowed in the base-layer. However, Short Header is allowed in the base-layer when FGS or FGS Temporal Scalability is not used in the Fine Granularity Scalable Object Type.

“

21) Add the following to the end of subclause 9.2:

"

Table AMD2-14 shows the definition of Advanced Simple Profile and Fine Granularity Scalable Profile.

Table AMD2-14 — Definition of Advanced Simple Profile and Fine Granularity Scalable Profile

ID	**Object Types** / **Profiles**	**Simple**	**Advanced Simple**	**Fine Granularity Scalable**
AS	Advanced Simple	X	X	
FGS	Fine Granularity Scalable	X	X	X

"

22) Add the following to Annex N after Table V2-44:

"

Within Advanced Simple Profile and Fine Granularity Scalable Profile, six levels are defined in each profile as specified in Table AMD2-15.

Table AMD2-15 — Definition of Levels in Advanced Simple Profile and Fine Granularity Scalable Profile

Visual Profile	Level	Typical Visual Session Size	Max. objects	Maximum number per type	Max. unique Quant Tables	Max. VMV buffer size (MB units)	Max VCV buffer size (MB)	VCV decoder rate (MB/s)	Max. Percentage of Intra MBs with AC prediction in VCV buffer	Max total VBV buffer size (units of 16384 bits)	Maximum VBV Buffer Size (units of 16384 bits)	Max. video packet length (bits)	Maximum Bitrate (kbits/s) (Note 2)	Maximum number of coded vop-bps (Note 3)
AS	L 0	176x144	1	1x AS or Simple	1	297	99	2970	100	10	10	2048	128	N.A.
AS	L 1	176x144	4	4x AS or Simple	1	297	99	2970	100	10	10	2048	128	N.A.
AS	L 2	352x288	4	4x AS or Simple	1	1188	396	5940	100	40	40	4096	384	N.A.
AS	L 3	352x288	4	4x AS or Simple	1	1188	396	11880	100	40	40	4096	768	N.A.
AS	L 4	352x576	4	4x AS or Simple	1	2376	792	23760	50	80	80	8192	3000	N.A.
AS	L 5	720x576	4	4x AS or Simple	1	4860	1620	48600	25	112	112	16384	8000	N.A.
FGS	L 0	176x144	1	1x AS or FGS or Simple	1	297	99	2970	100	10	10	2048	128	4

FGS	L1	176x144	4	4x AS or FGS or Simple	1	297	99	2970	100	10	10	2048	128	4
FGS	L2	352x288	4	4x AS or Simple	1	1188	396	5940	100	40	40	4096	384	4
FGS	L3	352x288	4	4x AS or FGS or Simple	1	1188	396	11880	100	40	40	4096	768	4
FGS	L4	352x576	4	4x AS or FGS or Simple	1	2376	792	23760	50	80	80	8192	3000	4
FGS	L5	720x576	4	4x AS or FGS or Simple	1	4860	1620	48600	25	112	112	16384	8000	4

NOTE 1 — The following restriction applies to L0 of AS Profile and FGS Profile:

- If AC prediction is used, QP value shall not be changed within a VOP (or within a video packet if video packets are used in a VOP). If AC prediction is not used, there are no restrictions to changing QP value.

NOTE 2 — For FGS Profile, this column is the maximum base-layer bitrate.

NOTE 3 — The maximum number of coded vop-bps takes into consideration the shifted bits after applying frequency weighting and/or selective enhancement.

NOTE 4 — The number of FGS, FGST, or FGS-FGST layer is always one. If FGS layer and FGST layer are separated, the number of total enhancement layers is two.

NOTE 5 — The interlace tools are not used for levels L0, L1, L2, and L3 of AS Profile and FGS Profile.

NOTE 6 — It is inherent in the FGS profile that the base and enhancement layers are tightly coupled to one another. To avoid unnecessary memory storage, the following constraints apply to the decoding time relationship of the enhancement layer and the base layer:

- Decoding and composition (or presentation in a no-compositor decoder) of each FGS or FGST VOP shall be performed in the same time unit.

- Decoding of each FGS and FGST VOP shall be performed immediately after the reference base layer VOP(s) are decoded without violating the above constraint.

"

23) Add the following clause B.4 after clause B.3:

"

B.4 Variable length codes for FGS

VLC Tables for FGS are specified in this Annex.

B.4.1 table_bpc

Table AMD2-16 — VLC table for MSB block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	3	101
3	2	0	4	1101
4	3	0	4	0001
5	4	0	5	11110
6	5	0	5	10011
7	6	0	6	110011
8	7	0	6	111010
9	8	0	6	100010
10	9	0	6	100101
11	10	0	7	1110011
12	11	0	7	0000001
13	12	0	8	11101101
14	13	0	8	10001111
15	14	0	9	110000101
16	15	0	9	111011100
17	16	0	9	000000001
18	17	0	9	000010001
19	18	0	9	000010000
20	19	0	10	1100000010
21	20	0	9	110000000
22	21	0	10	1000111010
23	22	0	11	11000011011
24	23	0	11	11101110110
25	24	0	11	10001110010
26	25	0	12	111011001100
27	26	0	12	000000000001
28	27	0	12	100011101100
29	28	0	13	1000111001110
30	29	0	13	1000111001111
31	30	0	14	00000000000001

32	31	0	13	1110010011000
33	32	0	13	1110010011010
34	33	0	13	1110010011011
35	34	0	13	1110010011001
36	35	0	13	0000000000001
37	0	1	3	001
38	1	1	5	11111
39	2	1	5	10000
40	3	1	6	111000
41	4	1	6	000011
42	5	1	6	000001
43	6	1	7	1100010
44	7	1	7	1100100
45	8	1	7	1100101
46	9	1	6	100100
47	10	1	7	1100011
48	11	1	7	1000110
49	12	1	7	0000101
50	13	1	8	11000001
51	14	1	8	11100101
52	15	1	8	00000001
53	16	1	9	000010011
54	17	1	9	110000111
55	18	1	9	111001000
56	19	1	9	110000100
57	20	1	8	11101111
58	21	1	9	111011000
59	22	1	9	000010010
60	23	1	10	1100001100
61	24	1	10	1100000011
62	25	1	10	1110111010
63	26	1	11	11000011010
64	27	1	11	11100100100
65	28	1	11	00000000010
66	29	1	11	00000000011
67	30	1	11	10001110111
68	31	1	11	11101100111
69	32	1	11	11100100111
70	33	1	11	11101110111
71	34	1	11	11100100101
72	35	1	10	1000111000
73	36	1	11	00000000001
74	37	1	12	111011001101
75	38	1	12	100011101101
76	39	1	12	100011100110
77	Escape	Code	10	1110110010

Table AMD2-17 — VLC table for MSB-1 block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	2	10
3	2	0	3	111
4	3	0	4	0010
5	4	0	4	1100
6	5	0	5	00010
7	6	0	5	11010
8	7	0	6	001110
9	8	0	6	000001
10	9	0	7	0011000
11	10	0	7	0001111
12	11	0	7	1101100
13	12	0	8	00011101
14	13	0	8	00000011
15	14	0	9	000110000
16	15	0	10	0011001111
17	16	0	10	0000000011
18	17	0	10	0000001001
19	18	0	11	00110011001
20	19	0	11	00110011011
21	20	0	11	00000000001
22	21	0	12	001100110000
23	22	0	12	000000101001
24	23	0	12	110111110000
25	24	0	13	0011001100011
26	25	0	13	0000001011101
27	26	0	13	1101111100010
28	27	0	15	000000000000001
29	0	1	6	001101
30	1	1	6	000011
31	2	1	7	0011111
32	3	1	7	0001101
33	4	1	7	0000100
34	5	1	7	0000101
35	6	1	7	1101101
36	7	1	8	00110010
37	8	1	8	00111100
38	9	1	8	00111101
39	10	1	8	00011100
40	11	1	8	00011001
41	12	1	8	00000001
42	13	1	8	11011110
43	14	1	9	000110001
44	15	1	10	0011001110
45	16	1	10	0000000010

46	17	1	10	0000000001
47	18	1	10	0000001000
48	19	1	10	1101111101
49	20	1	10	1101111111
50	21	1	11	00110011010
51	22	1	11	00000010110
52	23	1	11	11011111100
53	24	1	11	00000010101
54	25	1	11	11011111001
55	26	1	12	000000000001
56	27	1	12	000000101111
57	28	1	13	0000000000001
58	29	1	13	0000001010001
59	30	1	13	0000001011100
60	31	1	13	0000001010000
61	32	1	13	1101111100011
62	33	1	14	00000000000001
63	34	1	14	00110011000100
64	35	1	14	00110011000101
65	All	Zero	7	1101110
66	Escape	Code	11	11011111101

Table AMD2-18 — VLC table for MSB-2 block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	1	1
2	1	0	3	010
3	2	0	3	001
4	3	0	4	0111
5	4	0	5	01100
6	5	0	5	00001
7	6	0	6	011011
8	7	0	6	000101
9	8	0	7	0110100
10	9	0	7	0001110
11	10	0	8	00000101
12	11	0	8	00011001
13	12	0	9	000001000
14	13	0	10	0110101111
15	14	0	10	0000000001
16	15	0	11	00000100101
17	16	0	12	011010111011
18	17	0	12	000001001000
19	18	0	13	0110101110001
20	19	0	13	0110101101100
21	20	0	13	0110101101110
22	21	0	13	0001111000110

23	22	0	14	01101011100000
24	23	0	15	000000000000001
25	0	1	6	000100
26	1	1	7	0000001
27	2	1	7	0001101
28	3	1	8	01101010
29	4	1	8	00000110
30	5	1	8	00000001
31	6	1	8	00011111
32	7	1	8	00011000
33	8	1	9	000001111
34	9	1	9	000001110
35	10	1	9	000000001
36	11	1	9	000111101
37	12	1	10	0110101100
38	13	1	10	0000010011
39	14	1	10	0001111001
40	15	1	11	00000000001
41	16	1	12	011010111010
42	17	1	12	011010111001
43	18	1	12	000001001001
44	19	1	12	000000000001
45	20	1	12	000111100010
46	21	1	13	0110101101101
47	22	1	13	0110101101111
48	23	1	13	0000000000001
49	24	1	13	0001111000111
50	25	1	14	01101011100001
51	26	1	14	00000000000001
52	All	Zero	11	01101011010
53	Escape	Code	11	00011110000

Table AMD2-19 — VLC table for MSB-3 block-bp and other block-bps if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	1	1
2	1	0	2	01
3	2	0	3	001
4	3	0	5	00010
5	4	0	5	00001
6	5	0	6	000001
7	6	0	7	0001100
8	7	0	8	00011101
9	8	0	9	000111001
10	9	0	9	000000011
11	10	0	10	0000000010
12	11	0	11	00011010100

13	12	0	11	00000001010
14	13	0	12	000000000001
15	14	0	13	0000000011110
16	15	0	13	0000000101101
17	16	0	14	00000000000011
18	0	1	7	0001111
19	1	1	7	0000001
20	2	1	8	00011011
21	3	1	9	000111000
22	4	1	9	000110100
23	5	1	10	0001101011
24	6	1	10	0000000001
25	7	1	10	0000000100
26	8	1	11	00000000110
27	9	1	11	00000000001
28	10	1	12	000110101011
29	11	1	12	000000001110
30	12	1	12	000000010111
31	13	1	13	0000000011111
32	14	1	13	0000000101100
33	15	1	14	00000000000001
34	16	1	14	00000000000010
35	17	1	15	000000000000001
36	All	Zero	16	0000000000000001
37	Escape	Code	12	000110101010

The above VLC tables are implemented as the following arrays in C. The small array table_symbol[4][3] indicates the number of entries in the VLC tables for EOP=0, EOP=1, and ALL_ZERO. For example, the first row {36, 40, 0} indicates that the VLC table for the MSB block-bp has 36 RUN entries for EOP=0, 40 RUN entries for EOP=1, and no ALL_ZERO entry before the Escape Code. The second row {28, 36, 1} means that the VLC table for the MSB-1 block-bp has 28 RUN entries for EOP=0, 36 RUN entries for EOP=1, and 1 ALL_ZERO entry before the Escape Code. The large array table_bpc[4][154] contains the 4 VLC tables. The array contains all the VLC Codes and Code Lengths in the format of the decimal value of a VLC Code and its Code Length as a pair. For example, the first pair of numbers in table_bpc[4][154] is (1,2) that corresponds to the first VLC Code in the VLC Table for the MSB Block-bp, 01, with a Code Length 2. The second pair (5,3) corresponds to the second VLC Code, 101, in the same table with a Code Length 3.

```
int table_symbol[4][3] = {

{ 36,  40,  0},

{ 28,  36,  1},

{ 24,  27,  1},

{ 17,  18,  1},

};

int table_bpc[4][154] = {
```

{

1, 2, 5, 3, 13, 4, 1, 4, 30, 5, 19, 5, 51, 6, 58, 6, 34, 6, 37, 6, 115, 7, 1, 7, 237, 8, 143, 8, 389, 9, 476, 9, 1, 9, 17, 9, 16, 9, 770,10, 384, 9, 570,10, 1563,11, 1910,11, 1138,11, 3788,12, 1,12, 2284,12, 4558,13, 4559,13, 1,14, 7320,13, 7322,13, 7323,13, 7321,13, 1,13, 1, 3, 31, 5, 16, 5, 56, 6, 3, 6, 1, 6, 98, 7, 100, 7, 101, 7, 36, 6, 99, 7, 70, 7, 5, 7, 193, 8, 229, 8, 1, 8, 19, 9, 391, 9, 456, 9, 388, 9, 239, 8, 472, 9, 18, 9, 780,10, 771,10, 954,10, 1562,11, 1828,11, 2,11, 3,11, 1143,11, 1895,11, 1831,11, 1911,11, 1829,11, 568,10, 1,11, 3789,12, 2285,12, 2278,12, 946,10},

{

1, 2, 2, 2, 7, 3, 2, 4, 12, 4, 2, 5, 26, 5, 14, 6, 1, 6, 24, 7, 15, 7, 108, 7, 29, 8, 3, 8, 48, 9, 207,10, 3,10, 9,10, 409,11, 411,11, 1,11, 816,12, 41,12, 3568,12, 1635,13, 93,13, 7138,13, 1,15, 13, 6, 3, 6, 31, 7, 13, 7, 4, 7, 5, 7, 109, 7, 50, 8, 60, 8, 61, 8, 28, 8, 25, 8, 1, 8, 222, 8, 49, 9, 206,10, 2,10, 1,10, 8,10, 893,10, 895,10, 410,11, 22,11, 1788,11, 21,11, 1785,11, 1,12, 47,12, 1,13, 81,13, 92,13, 80,13, 7139,13, 1,14, 3268,14, 3269,14, 110, 7, 1789,11},

{

1, 1, 2, 3, 1, 3, 7, 4, 12, 5, 1, 5, 27, 6, 5, 6, 52, 7, 14, 7, 5, 8, 25, 8, 8, 9, 431,10, 1,10, 37,11, 1723,12, 72,12, 3441,13, 3436,13, 3438,13, 966,13, 6880,14, 1,15, 4, 6, 1, 7, 13, 7, 106, 8, 6, 8, 1, 8, 31, 8, 24, 8, 15, 9, 14, 9, 1, 9, 61, 9, 428,10, 19,10, 121,10, 1,11, 1722,12, 1721,12, 73,12, 1,12, 482,12, 3437,13, 3439,13, 1,13, 967,13, 6881,14, 1,14, 858,11, 240,11},

{

1, 1, 1, 2, 1, 3, 2, 5, 1, 5, 1, 6, 12, 7, 29, 8, 57, 9, 3, 9, 2,10, 212,11, 10,11, 1,12, 30,13, 45,13, 3,14, 15, 7, 1, 7, 27, 8, 56, 9, 52, 9, 107,10, 1,10, 4,10, 6,11, 1,11, 427,12, 14,12, 23,12, 31,13, 44,13, 1,14, 2,14, 1,15, 1,16, 426,12}

};

B.4.2 table_bpc4fw_m

Table AMD2-20 — VLC table for MSB block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	10
2	1	0	4	0110
3	2	0	5	01110
4	3	0	6	011110
5	4	0	6	010111
6	5	0	6	000011
7	6	0	6	000110
8	7	0	7	0101010
9	8	0	7	0101000
10	9	0	7	0000001
11	10	0	8	01111111
12	11	0	8	01011001
13	12	0	8	00010111
14	13	0	9	011111000
15	14	0	9	010101101
16	15	0	9	000000001

17	16	0	10	0111110011
18	17	0	10	0111110010
19	18	0	10	0101011110
20	19	0	10	0101101101
21	20	0	11	00000000001
22	0	1	2	11
23	1	1	3	001
24	2	1	4	0100
25	3	1	6	000100
26	4	1	7	0101001
27	5	1	7	0000010
28	6	1	7	0001010
29	7	1	7	0001110
30	8	1	7	0001111
31	9	1	8	01111101
32	10	1	8	01111110
33	11	1	8	01011000
34	12	1	8	00000001
35	13	1	8	00000110
36	14	1	8	00010110
37	15	1	9	010101100
38	16	1	9	010110111
39	17	1	9	010101110
40	18	1	9	010110101
41	19	1	9	010110100
42	20	1	9	000001110
43	21	1	10	0101011111
44	22	1	10	0000011111
45	23	1	10	0000000001
46	24	1	10	0000011110
47	25	1	10	0101101100
48	Escape	Code	6	000010

Table AMD2-21 — VLC table for MSB-1 block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	3	001
3	2	0	3	100
4	3	0	4	1101
5	4	0	5	00001
6	5	0	5	11100
7	6	0	6	000110
8	7	0	6	000001
9	8	0	6	110001
10	9	0	7	0001111
11	10	0	7	1110111
12	11	0	7	1010010

13	12	0	8	11101001
14	13	0	8	11000010
15	14	0	9	000111000
16	15	0	9	110000111
17	16	0	9	101010100
18	17	0	10	0000000011
19	18	0	10	1110110111
20	19	0	10	1100001101
21	20	0	10	1010111010
22	21	0	11	00000000001
23	22	0	11	11000011001
24	23	0	11	10101010111
25	24	0	12	000111001010
26	0	1	4	1111
27	1	1	5	00010
28	2	1	5	11001
29	3	1	6	101000
30	4	1	7	0000001
31	5	1	7	1110101
32	6	1	7	1010100
33	7	1	7	1010110
34	8	1	7	1010011
35	9	1	8	00011101
36	10	1	8	00000001
37	11	1	8	11101000
38	12	1	8	11101100
39	13	1	8	11000000
40	14	1	8	10101111
41	15	1	9	111011010
42	16	1	9	101010111
43	17	1	9	101010110
44	18	1	9	101011100
45	19	1	10	0000000010
46	20	1	10	0000000001
47	21	1	10	1110110110
48	22	1	10	1010101010
49	23	1	10	1010111011
50	24	1	11	00011100100
51	25	1	11	00011100110
52	26	1	11	11000011000
53	27	1	11	10101010110
54	28	1	12	000111001011
55	29	1	12	000111001111
56	30	1	12	000111001110
57	31	1	12	000000000001
58	32	1	13	0000000000001
59	All	Zero	4	1011
60	Escape	Code	8	11000001

Table AMD2-22 — VLC table for MSB-2 block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	2	10
3	2	0	3	110
4	3	0	4	0001
5	4	0	4	1110
6	5	0	5	00101
7	6	0	5	11110
8	7	0	6	000001
9	8	0	6	111111
10	9	0	7	0000001
11	10	0	7	0011010
12	11	0	8	00001100
13	12	0	8	00110010
14	13	0	9	000010010
15	14	0	9	001101101
16	15	0	10	0000100111
17	16	0	10	0000000001
18	17	0	11	00001011111
19	18	0	11	00000000100
20	19	0	11	00000000001
21	20	0	11	00110110000
22	21	0	12	000011111011
23	22	0	12	001001010010
24	23	0	13	0000101111010
25	24	0	13	0000100110010
26	25	0	14	00000000000001
27	0	1	5	00111
28	1	1	6	001000
29	2	1	6	111110
30	3	1	7	0010011
31	4	1	7	0011000
32	5	1	8	00001010
33	6	1	8	00001110
34	7	1	8	00001101
35	8	1	8	00000001
36	9	1	8	00100100
37	10	1	8	00110011
38	11	1	8	00110111
39	12	1	9	000010110
40	13	1	9	000011110
41	14	1	9	001001011
42	15	1	10	0000111111
43	16	1	10	0000000011
44	17	1	10	0010010101
45	18	1	10	0011011001